

# Desenvolvimento de Serviços e APIs

Curso Superior de Tecnologia em Análise  
e Desenvolvimento de Sistemas  
Prof. Edécio Fernando Iepsen



# Vamos falar sobre...

1

Plano de Ensino

2

O que são APIs

3

Tecnologias  
Utilizadas

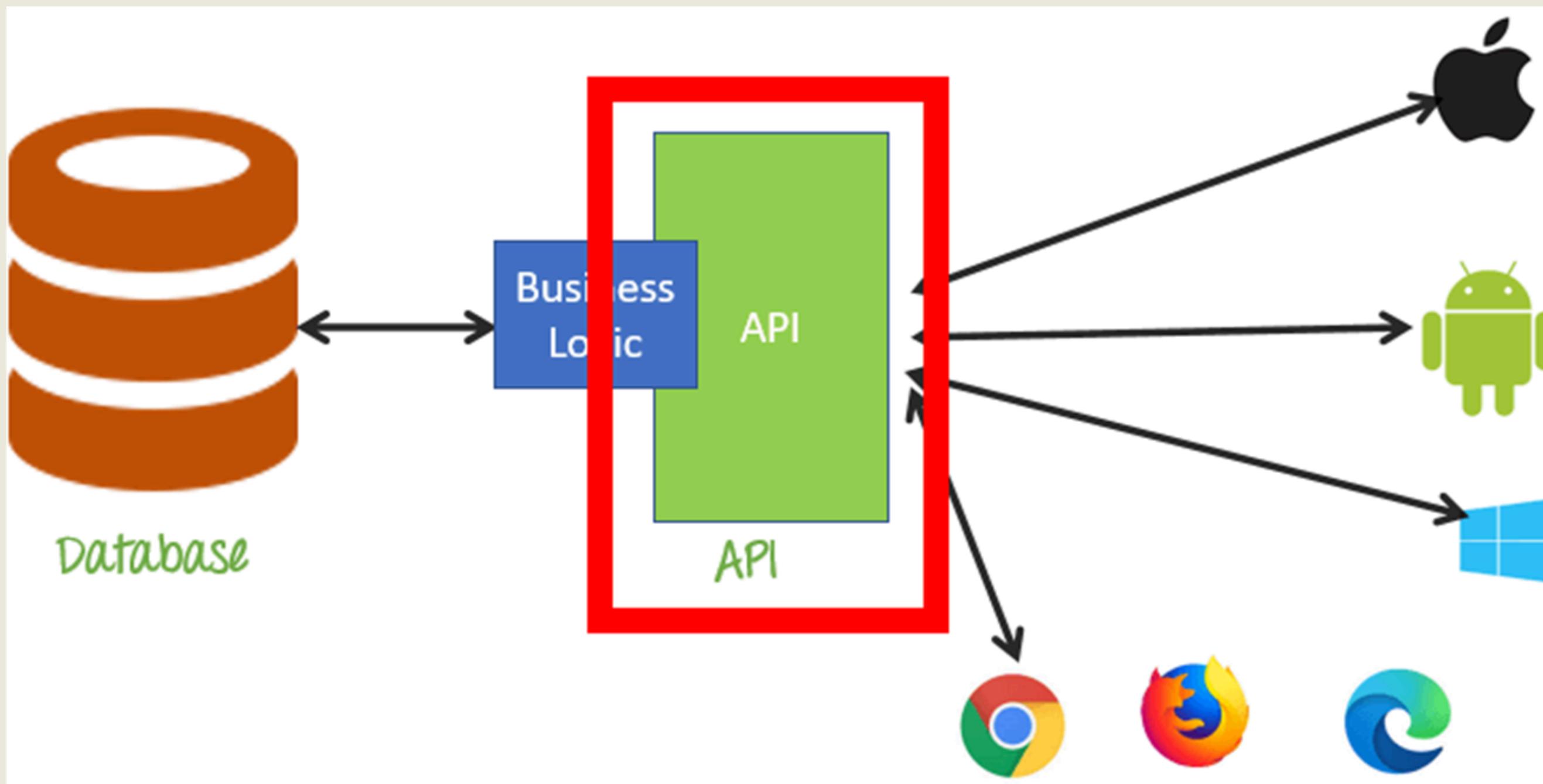
4

Conceitos  
Básicos

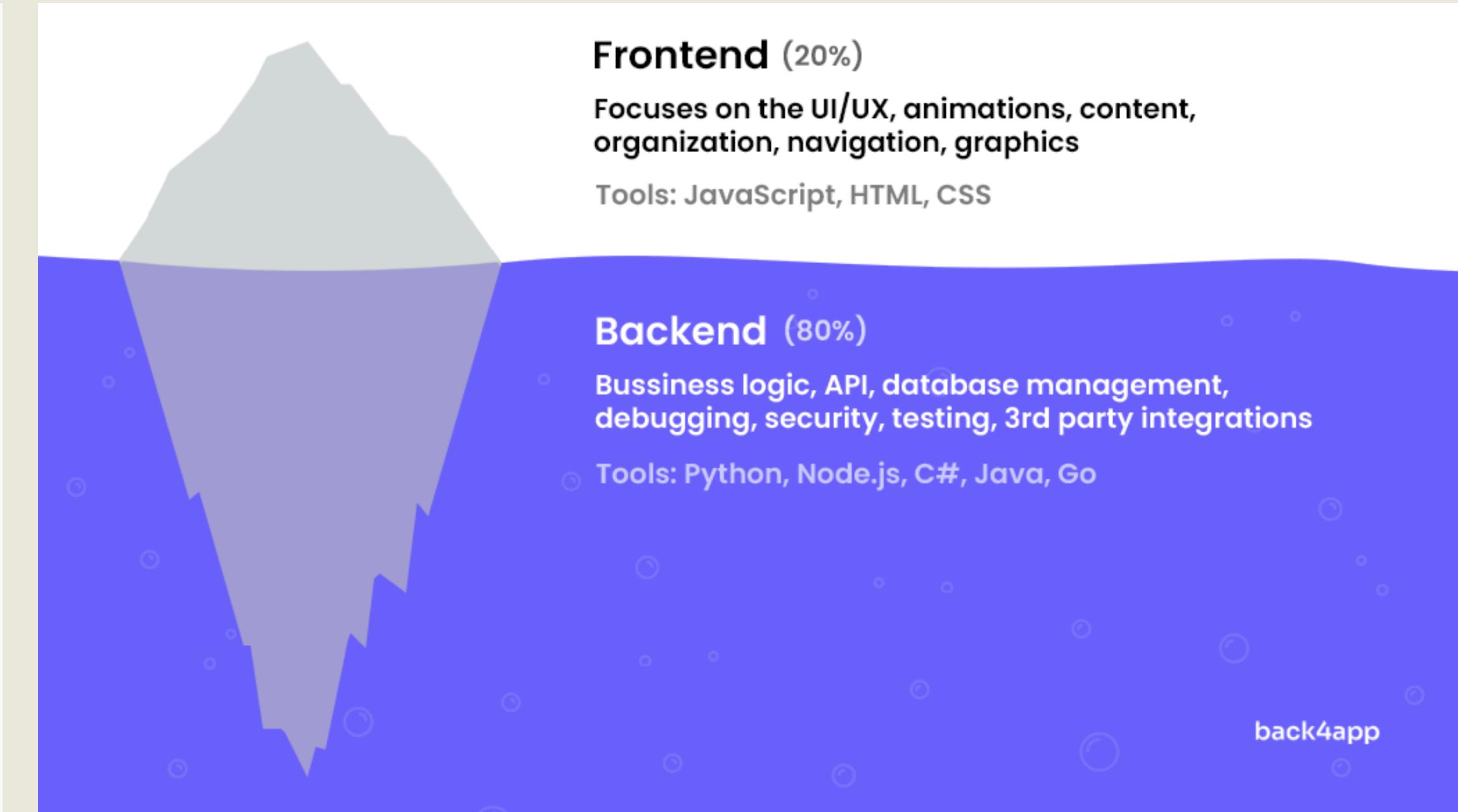
5

Exemplos e  
Exercícios

# API (*Application Programming Interface*) - Interface de Programação de Aplicações



- Desenvolvimento Back-End (Node js / Express js)
- Desenvolvimento Front-End (Reactjs, Next js)



# Tecnologia: estudo mostra tendências de trabalho para 2025

O balanço aponta quais são as principais oportunidades para vagas efetivas e por projetos; Cristina Boner traça perspectivas do mercado de trabalho de tecnologia para o próximo ano

17 dez 2024 - 12h23

Compartilhar



[Exibir comentários](#)



Uma sondagem divulgada pela [Robert Half](#) revelou quais são as principais tendências de contratação e salários do setor de tecnologia para 2025. Os cargos mais demandados em tecnologia para vagas efetivas são Gerente de Tecnologia da Informação (TI) Generalista, Desenvolvedor Back-end, Coordenador de Segurança da Informação e Gerente de Projetos.



Imagen: seventyfourimages / envato

Os cargos de gerente de TI, desenvolvedor back-end e coordenador de segurança da informação serão os cargos mais promissores em 2025, **com remuneração inicial entre R\$ 9 mil e R\$ 20 mil**. É o que mostra a nova edição do guia salarial anual feito pela recrutadora Robert Half, que traz um panorama de tendências para a área de **tecnologia** no próximo ano.

**Por meio de APIs,  
desenvolvedores  
podem criar  
novos softwares  
e aplicativos  
capazes de se  
comunicar com  
outras  
plataformas.**

**O objetivo de uma  
API é simplificar o  
desenvolvimento  
de programas, a  
partir da  
abstração dos  
diversos elementos  
que o compõem.**

**Um desenvolvedor  
não precisa saber  
em detalhes o  
funcionamento das  
APIs, mas sim, em  
como utilizá-las e  
como se dará a  
interação dela com  
o seu software.**

Search...

Campos retornados

Authentication

Consultar CEP

[Documentation Powered by ReDoc](#)

Authentication

Consultar CEP

[GET Consulta CEP](#)[Documentation Powered by ReDoc](#)

# API Consulta CEP (1.0.2)

Download OpenAPI specification: [Download](#)

A API [Consulta CEP](#) é disponibilizada através de serviço [HTTP REST](#). O objetivo da API Consulta CEP é disponibilizar um determinado endereço a partir do CEP

## Consulta CEP

Apresenta o endereço referente a um CEP

### Exemplo

#### Chamada

```
https://h-apigateway.conectagov.estaleiro.serpro.gov.br/api-cep/v1/consulta/cep/60130240
```

#### Retorno

```
{
  "cep": "60130240",
  "tipoCep": "logradouro",
  "subTipoCep": "S",
  "uf": "CE",
  "cidade": "Fortaleza",
  "bairro": "São João do Tauape",
  "endereco": "Avenida Pontes Vieira",
  "complemento": "De 2 Até 1550 Lado Par",
  "codigoIBGE": ""
```

## Frontend



## Backend

## Frontend

Check the menu and make the order



## Backend



Kitchen

API

# Integração

Uma API é um conjunto de normas que possibilita a comunicação ou integração entre softwares/plataformas a partir de uma série de padrões e protocolos.



# **Web Service ou API Web:**

**É uma API no contexto do desenvolvimento Web.**

**São um conjunto de mensagens de requisição e resposta HTTP, geralmente expresso nos formatos XML ou JSON.**

**Enquanto você usufrui de um aplicativo ou site, este pode estar conectado a diversos outros sistemas e aplicativos via APIs sem que se perceba.**

**Um exemplo popular é a rede social Twitter, sendo possível ler e publicar mensagens.**





# App Minha Lista de Filmes

## Cadastro e Avaliação Pessoal de Filmes

### Listas dos Filmes Cadastrados

[Novo Filme](#)

**Amigos Imaginários**  
Comédia - 108 min  
Uma garota descobre que consegue ver os amigos imaginários de todas as pessoas, mesmo aqueles esquecidos por crianças que já cresceram.

**NOVIDADE**

[Avaliar ✓](#)

**Garfield: Fora de Casa**  
Animação - 102 min  
Garfield tem uma aventura selvagem com seu pai perdido, Vic, e seu amigo Odie, em um assalto hilariante e arriscado.

**★★★★★**  
Bom para ver com as crianças. Gostei

**Planeta dos Macacos**  
Ação - 138 min  
Várias gerações após o reinado de César, os macacos são a espécie dominante, vivendo harmoniosamente, e os humanos foram reduzidos a viver nas sombras.

**★★★★★**  
História tradicional, porém, com muita ação!

**Divertidamente 2**  
Animação - 96 min  
A sede dos sentimentos (mente) de Riley está passando por uma demolição repentina para dar lugar a novas emoções!

**★★★★★**

**Bad Boys: Até o Fim**  
Ação - 115 min  
Os Bad Boys preferidos do mundo todo estão de volta com a mistura icônica de ação eletrizante e comédia

**Assassino por Acaso**  
Ação - 115 min  
Um policial disfarçado de assassino de aluguel quebra o protocolo para tentar salvar uma mulher desesperada.



# App Minha Lista de Filmes

## Cadastro e Avaliação Pessoal de Filmes

### Lista dos Filmes Cadastrados



**Amigos  
Imaginários**

**Comédia - 108  
min**

Uma garota descobre que consegue ver os amigos imaginários de todas as pessoas, mesmo aqueles esquecidos por crianças que já cresceram.

**NOVIDADE**

**Avaliar ✓**



**Garfield:  
Fora de  
Casa**

**Animação  
- 102 min**

Garfield tem uma aventura selvagem com seu pai perdido, Vic, e seu amigo Odie, em um assalto hilariante e arriscado.



Bom para

**Novo Filme**

Aplicativo

- Manifesto
- Service workers
- Armazenamento

Armazenamento

- Armazenamento local
  - http://localhost:5173
- Armazenamento da sessão
- IndexedDB
- Cookies
  - http://localhost:5173
- Tokens de estado particular
- Grupos de interesse
- Armazenamento compartilhado
- Armazenamento em cache
- Buckets de armazenamento

Serviços em segundo plano

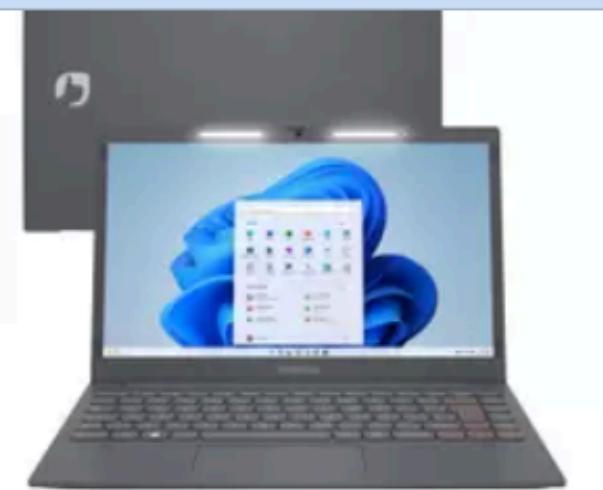
- Cache de avanço e retorno
- Busca em segundo plano
- Sincronização em segundo plano
- Mitigações de rastreio por recomendação
- Notificações
- Gerenciador de pagamentos
- Sincronização periódica em segundo plano
- Carregamentos especulativos
- Mensagens push

Origem http://localhost:5173

Chave	Valor
filmes	[{"titulo": "Amigos Imaginários", "genero": "Comédia", "duracao": 108, ...}, {"titulo": "Garfield: Fora de Casa", "genero": "Animação", "duracao": 102, ...}, {"titulo": "Planeta dos Macacos", "genero": "Ação", "duracao": 138, ...}, {"titulo": "Divertidamente 2", "genero": "Animação", "duracao": 96, ...}, {"titulo": "Bad Boys: Até o Fim", "genero": "Ação", "duracao": 115, ...}, {"titulo": "Assassino por Acaso", "genero": "Ação", "duracao": 115, ...}, {"titulo": "Furiosa: Uma Saga Mad Max", "genero": "Ação", "duracao": 148, ...}]

Console Novidades Problemas

- 5
- 4 e acima
- 3 e acima
- 2 e acima
- 1 e acima



## Vendido por

- Magalu
- Netshoes
- KaBuM!
- Equipatech
- Bringitsc2

[Ver todos](#)

## Tipo de produto

- Notebook
- Bateria para Notebook
- Fonte de Alimentação
- Teclado
- Tela para Notebook

[Ver todos](#)

▲ Notebook Positivo Vision C14 Intel Celeron 4GB  
 110  
R\$ 1.699,00  
**R\$ 1.151,10** no Pix  
(10% de desconto)  
ou R\$ 1.279,00 em 10x de R\$ 127,90 sem juros

Notebook Asus Vivobook 15 Intel Core i3 4GB 256GB  
 254  
R\$ 3.799,00  
**R\$ 2.024,10** no Pix  
(10% de desconto)  
ou R\$ 2.249,00 em 10x de R\$ 224,90 sem juros

Notebook Compaq Presario Snapdragon 7c 4GB  
 106  
R\$ 2.199,00  
**R\$ 1.160,10** no Pix  
(10% de desconto)  
ou R\$ 1.289,00 em 10x de R\$ 128,90 sem juros



▲ Notebook HP Intel Core i3 8GB 256GB SSD 15,6"  
 328  
R\$ 3.999,00  
**R\$ 2.249,10** no Pix



Notebook Asus Vivobook 15 Intel Core i3 4GB 256GB  
 93  
R\$ 3.499,00  
**R\$ 1.889,10** no Pix



**MAIS VENDIDO**  
Notebook Acer Aspire 5 Intel Core i5H 8GB 512GB SS  
 257  
R\$ 6.416,86  
**R\$ 2.789,10** no Pix

# Linguagens / Frameworks Back-end (Full Stack)



Express 

The text "Express" is in a black, lowercase, sans-serif font. To its right is a yellow square containing the letters "JS" in a bold, black, sans-serif font.

Começando

## Introdução ao Node.js

Como instalar o  
Node.js

Quanto JavaScript  
você precisa saber  
para usar o Node.js?

Diferenças entre  
Node.js e o navegador

O mecanismo  
JavaScript V8

Uma introdução ao  
gerenciador de  
pacotes npm

ECMAScript 2015 (ES6)  
e além

Node.js, a diferença  
entre  
desenvolvimento e  
produção

Node.js com  
TypeScript

Node.js com  
WebAssembly

Trabalho Assíncrono

# Introdução ao Node.js

Node.js é um ambiente de tempo de execução JavaScript de código aberto e plataforma cruzada. É uma ferramenta popular para quase qualquer tipo de projeto!

O Node.js executa o mecanismo JavaScript V8, o núcleo do Google Chrome, fora do navegador. Isso permite que o Node.js tenha muito desempenho.

Um aplicativo Node.js é executado em um único processo, sem criar um novo thread para cada solicitação. O Node.js fornece um conjunto de primitivas de E/S assíncronas em sua biblioteca padrão que evitam o bloqueio do código JavaScript e, geralmente, as bibliotecas no Node.js são escritas usando paradigmas sem bloqueio, tornando o comportamento de bloqueio a exceção e não a norma.

Quando o Node.js executa uma operação de E/S, como ler da rede, acessar um banco de dados ou sistema de arquivos, em vez de bloquear o thread e desperdiçar ciclos de CPU esperando, o Node.js retomará as operações quando a resposta voltar.

Isso permite que o Node.js lide com milhares de conexões simultâneas com um único servidor sem introduzir a carga de gerenciamento da simultaneidade de threads, o que pode ser uma fonte significativa de bugs.

# Express

4.18.1

Fast, unopinionated, minimalist  
web framework for [Node.js](#)

```
$ npm install express --save
```

## Aplicativos da web

Express é uma estrutura de aplicativo da web Node.js mínima e flexível que fornece um conjunto robusto de recursos para aplicativos da web e móveis.

## APIs

Com uma infinidade de métodos utilitários HTTP e middleware à sua disposição, criar uma API robusta é rápido e fácil.

## Desempenho

O Express fornece uma camada fina de recursos fundamentais de aplicativos da Web, sem obscurecer os recursos do Node.js que você conhece e adora.

## Estruturas

Muitas [estruturas populares](#) são baseadas no Express.

# TypeScript



Visão geral | Aprendendo  
Orientação a Objetos com TS #1



Todo Dev JavaScript Precisa do TypeScript

29 mil visualizações • há 3 meses

 Código Fonte TV ✓

TypeScript trouxe o Open Source para



# Criar um novo projeto

```
C:\aulas24\api>md aula1  
  
C:\aulas24\api>cd aula1  
  
C:\aulas24\api\aula1>npm init -y  
Wrote to C:\aulas24\api\aula1\package.json:
```

# Criar um novo projeto: dependências

```
C:\aulas24\api\aula1>npm i typescript ts-node-dev --save-dev
```

npm i typescript ts-node-dev --save-dev

```
C:\aulas24\api\aula1>npm i @types/node @types/express --save-dev
```

npm i @types/node @types/express --save-dev

```
C:\aulas24\api\aula1>npm i express
```

npm i express

# Inicializar Arquivo de Dependências do TS

```
C:\apis_251\manha\aula1>npx tsc --init
```

npx tsc --init

Gera um arquivo (tsconfig.json) com diversas configurações que controlam como o TypeScript compila o código .ts

# Script: dev

```
C: > aulas24 > api > aula1 > {} package.json > ...
1  {
2    "name": "aula1",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    ▷ Debug
7    "scripts": {
8      "dev": "npx ts-node-dev --respawn index.ts"
9    },
10   "keywords": [],
11   "author": ""
```

npx ts-node-dev --respawn index.ts

The screenshot shows the npmjs.com package page for 'ts-node-dev'. At the top, there are navigation icons for back, forward, and refresh, followed by the URL 'npmjs.com/package/ts-node-dev'. Below the header is the npm logo and a search bar labeled 'Search packages'. The main content area features the package name 'ts-node-dev' in large bold letters, its version '2.0.0', and its status as 'Public' and 'Published 2 years ago'. There are three tabs: 'Readme' (selected), 'Code' (with a 'Beta' badge), and '10 Dependencies'. A summary box states: 'Tweaked version of node-dev that uses ts-node under the hood.' Below this, a detailed description explains that it restarts the target node process when required files change, sharing the TypeScript compilation process between restarts, which increases speed compared to other methods like nodemon.

## ts-node-dev

2.0.0 • Public • Published 2 years ago

[Readme](#)

[Code](#) Beta

[10 Dependencies](#)

## ts-node-dev

Tweaked version of [node-dev](#) that uses [ts-node](#) under the hood.

It restarts target node process when any of required files changes (as standard `node-dev`) but shares [TypeScript](#) compilation process between restarts. This significantly increases speed of restarting comparing to `node-dev -r ts-node/register ...`, `nodemon -x ts-node ...` variations because there is no need to instantiate `ts-node` compilation each time.

Black Lives Matter.

Support the Equal Justice Initiative.

Express

search

Página Inicial

Introdução

Guia

Referência da API

Tópicos Avançados

Recursos

## Exemplo Hello World

Este é essencialmente o aplicativo mais simples do Express que é possível criar. Ele é um aplicativo de arquivo único — **não** é o que você iria obter usando o [Gerador Express](#), que cria a estrutura para um aplicativo completo com inúmeros arquivos JavaScript, modelos Jade, e subdiretórios para vários propósitos.

Primeiro crie um diretório chamado `myapp`, mude para ele e execute o `npm init`. Em seguida instale o `express` como uma dependência, de acordo com o [guia de instalação](#).

No diretório `myapp`, crie um arquivo chamado `app.js` e inclua o seguinte código:

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

O aplicativo inicia um servidor e escuta a porta 3000 por conexões. O aplicativo responde com "Hello World!" à solicitações para a URL raiz `/` ou **rota**. Para todos os outros caminhos, ele irá responder com um **404 Não Encontrado**.

TS index.ts X

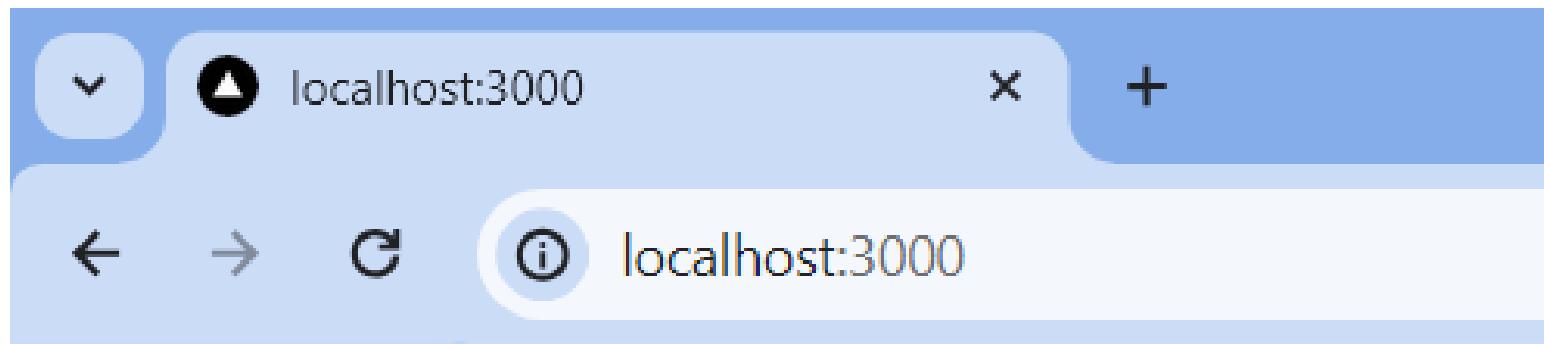
TS index.ts > ...

```
1 import express from 'express'
2 const app = express()
3 const port = 3000
4
5 app.get('/', (req, res) => {
6   res.send('Aula 1: APIs')
7 })
8
9 app.listen(port, () => {
10   console.log(`Servidor rodando na porta ${port}`)
11 })
12
```

```
C:\apis24_2\manha\aula1>npm run dev
```

```
> aula1@1.0.0 dev
> npx ts-node-dev --respawn index.ts
```

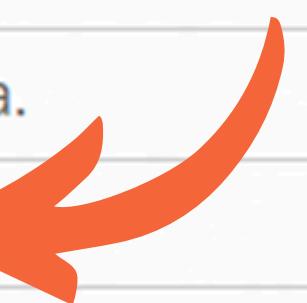
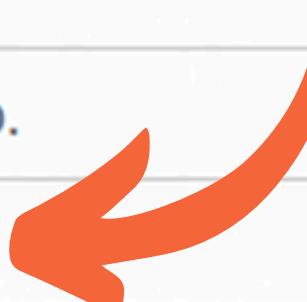
```
[INFO] 09:58:50 ts-node-dev ver. 2.0.0 (using ts-node ver. 10.9.2, typescript ver. 5.5.4)
Servidor rodando na porta: 3000
```



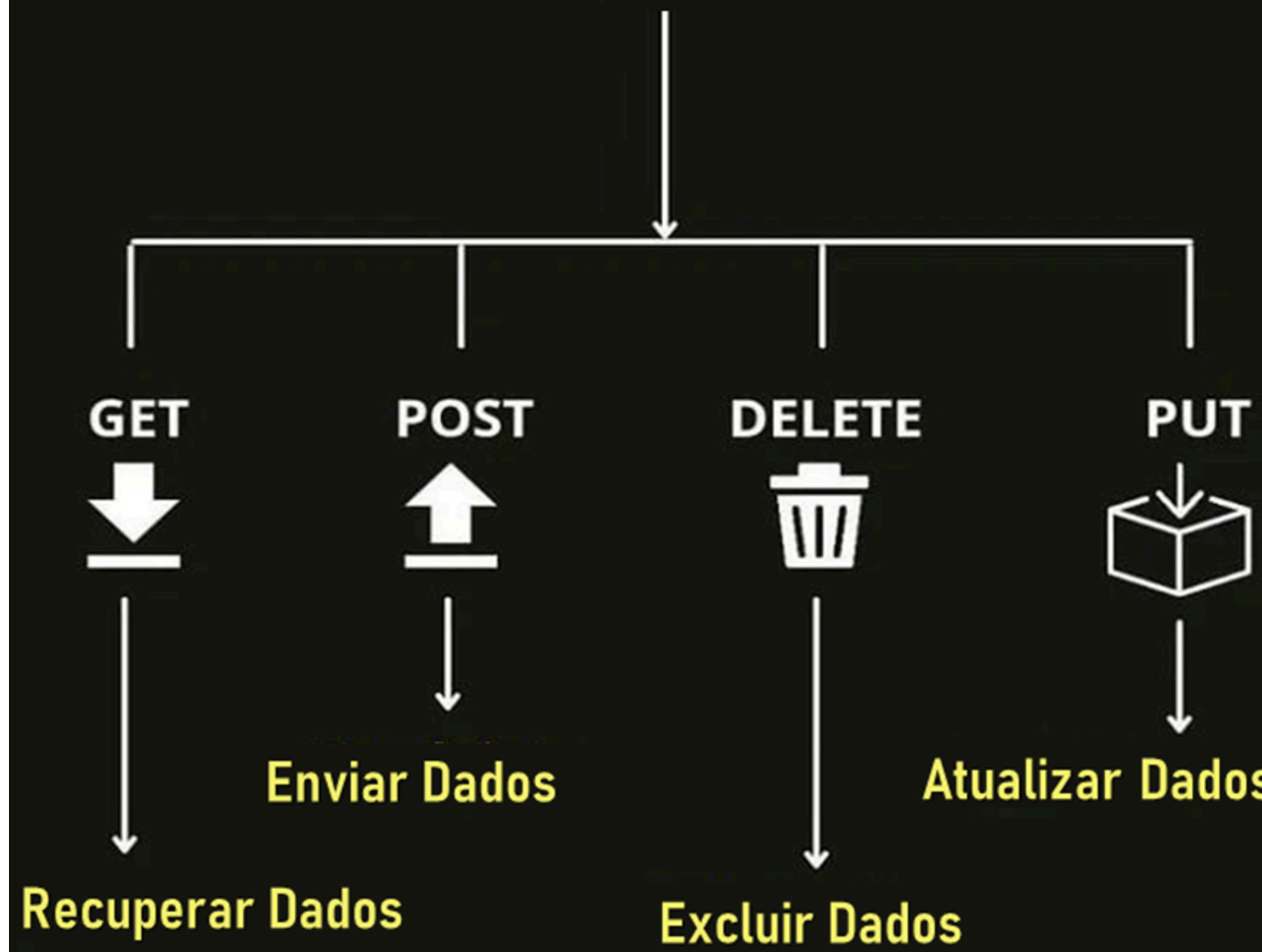
Aula 1: APIs

## Métodos de resposta

Os métodos do objeto de resposta (res) na seguinte tabela podem enviar uma resposta ao cliente, e finalizar o ciclo solicitação-resposta. Se nenhum destes métodos forem chamados a partir de um manipulador de rota, a solicitação do cliente será deixada em suspenso.

Método	Descrição
<code>res.download()</code>	Solicita que seja efetuado o download de um arquivo
<code>res.end()</code>	Termina o processo de resposta.
<code>res.json()</code>	Envia uma resposta JSON. 
<code>res.jsonp()</code>	Envia uma resposta JSON com suporte ao JSONP.
<code>res.redirect()</code>	Redireciona uma solicitação.
<code>res.render()</code>	Renderiza um modelo de visualização.
<code>res.send()</code>	Envia uma resposta de vários tipos. 
<code>res.sendFile</code>	Envia um arquivo como um fluxo de octeto.
<code>res.sendStatus()</code>	Configura o código do status de resposta e envia a sua representação em sequência de caracteres como o corpo de resposta.

# PRINCIPAIS MÉTODOS HTTP



# Basic routing

**Routing** refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

Each route can have one or more handler functions, which are executed when the route is matched.

Route definition takes the following structure:

```
app.METHOD(PATH, HANDLER)
```

Where:

- app is an instance of express.
- METHOD is an [HTTP request method](#), in lowercase.
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

The following examples illustrate defining simple routes.

Respond with Hello World! on the homepage:

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

Respond to POST request on the root route (/), the application's home page:

```
app.post('/', (req, res) => {
  res.send('Got a POST request')
})
```

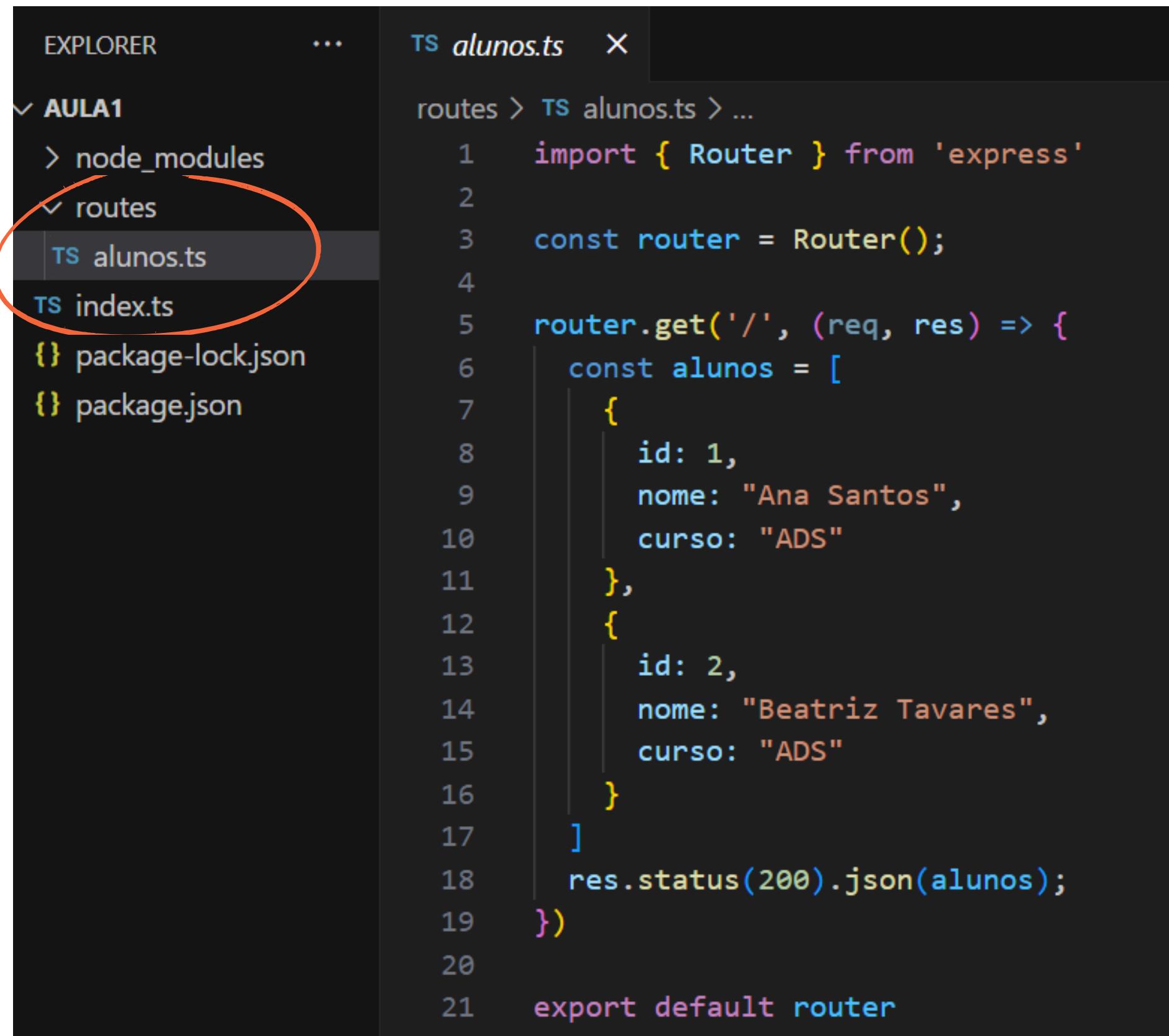
Respond to a PUT request to the /user route:

```
app.put('/user', (req, res) => {
  res.send('Got a PUT request at /user')
})
```

Respond to a DELETE request to the /user route:

```
app.delete('/user', (req, res) => {
  res.send('Got a DELETE request at /user')
})
```

# Exemplo de API (ainda sem BD)



The screenshot shows a Visual Studio Code interface. The left sidebar (EXPLORER) displays a file tree for a project named 'AULA1'. The 'routes' folder and its file 'alunos.ts' are highlighted with a red oval. Other files visible include 'node\_modules', 'index.ts', 'package-lock.json', and 'package.json'. The right pane (EDITOR) shows the code for 'alunos.ts'. The code defines an Express router that handles a GET request to the root path ('/') and returns a JSON response containing two student objects.

```
TS alunos.ts
routes > TS alunos.ts > ...
1 import { Router } from 'express'
2
3 const router = Router();
4
5 router.get('/', (req, res) => {
6   const alunos = [
7     {
8       id: 1,
9       nome: "Ana Santos",
10      curso: "ADS"
11    },
12    {
13      id: 2,
14      nome: "Beatrix Tavares",
15      curso: "ADS"
16    }
17  ]
18  res.status(200).json(alunos);
19 }
20
21 export default router
```

# HTTP Status Codes



## HTTP Status Codes

### Level 200

200: OK  
201: Created  
202: Accepted  
203: Non-Authoritative Information  
204: No content

### Level 400

400: Bad Request  
401: Unauthorized  
403: Forbidden  
404: Not Found  
409: Conflict

### Level 500

500: Internal Server Error  
501: Not Implemented  
502: Bad Gateway  
503: Service Unavailable  
504: Gateway Timeout  
599: Network Timeout

# Importar arquivo com rotas

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows a project structure with a folder named "AULA1" containing "node\_modules", "routes" (which contains "alunos.ts"), and "index.ts".
- index.ts** file (selected in the Explorer):

```
1 import express from 'express'
2
3 import alunosRoutes from './routes/alunos'
4
5 const app = express();
6 const port = 3000;
7
8 app.use(express.json())
9 app.use('/alunos', alunosRoutes)
10
11 app.get('/', (req, res) => {
12   res.send('Aula 1: APIs');
13 })
14
15 app.listen(port, () => {
16   console.log(`Servidor rodando em http://localhost:${port}`);
17 })
18
```

A red bracket highlights the import statement "import alunosRoutes from './routes/alunos'" and the closing brace of the module export at the bottom of the code.

A screenshot of a web browser window displaying a JSON array. The browser's address bar shows the URL `localhost:3000/alunos`. The main content area contains the following JSON data:

```
1 // 20240211101118
2 // http://localhost:3000/alunos
3
4 [
5   {
6     "id": 1,
7     "nome": "Ana Santos",
8     "curso": "ADS"
9   },
10  {
11    "id": 2,
12    "nome": "Ana Santos",
13    "curso": "ADS"
14  }
15 ]
```

The JSON structure represents two student records. Each record has an `id`, `nome`, and `curso` field. The browser interface includes a gear icon for settings, a raw file icon, and navigation controls.

# Exemplo de API (ainda sem BD)

```
21
22  router.post('/', (req, res) => {
23    const { nome, curso } = req.body
24    alunos.push(
25      {
26        id: alunos.length + 1,
27        nome,
28        curso
29      })
30    res.status(201).json({ msg: "Aluno inserido com sucesso", id: alunos.length })
31  })
32
33  export default router
34
```

## req.body

Contains key-value pairs of data submitted in the request body. By default, it is `undefined`, and is populated when you use body-parsing middleware such as `express.json()` or `express.urlencoded()`.

The following example shows how to use body-parsing middleware to populate `req.body`.

```
var express = require('express')

var app = express()

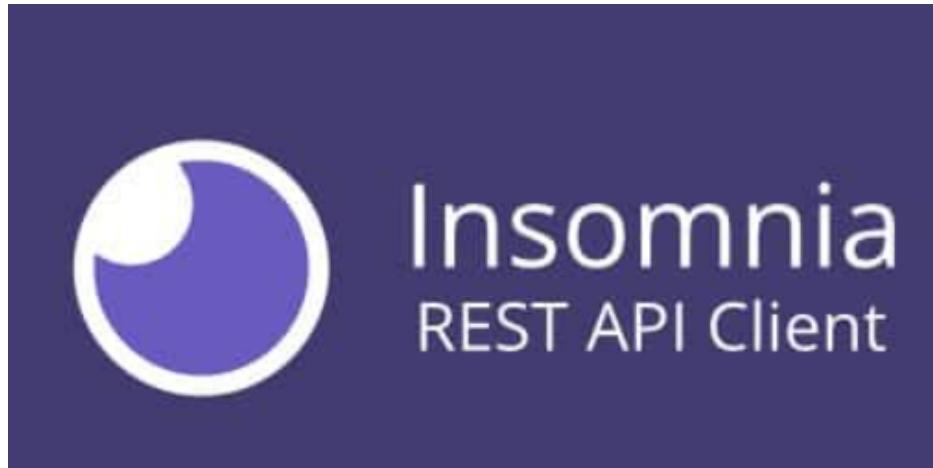
app.use(express.json()) // for parsing application/json
app.use(express.urlencoded({ extended: true })) // for parsing application/x-www-form-urlencoded

app.post('/profile', function (req, res) {
  console.log(req.body)
  res.json(req.body)
})
```

# Testes das Rotas (softwares)



POSTMAN



The screenshot shows the Insomnia download page at [insomnia.rest/download](https://insomnia.rest/download). The page has a dark theme with a purple header. It features the Insomnia logo and navigation links for Products, Docs, Pricing, Plugins, and Login. A prominent "Get Started for Free →" button is visible. On the right, a large window displays the Insomnia application's interface. The interface includes a "DOCUMENT" section with environment, cookie, and certificate settings. Below it is an "ENDPOINTS" tree view showing API endpoints like /pet, /store, and /user, each with corresponding HTTP methods (e.g., GET, POST, DELETE). To the right of the tree, a "POST" request is selected with a URL of "base\_url /pet". The "JSON" tab is active, showing a JSON schema for a pet object. The schema includes properties like id, category, name, photoUrls, tags, and status. A "Send" button is at the top right of the request panel. At the bottom of the interface, there are "Settings" and "Beautify JSON" buttons.

# Insomnia

The image displays three sequential screenshots of the Insomnia 9 welcome screen:

- Screenshot 1:** Shows a list of notable features: Pre-request scripting, Native API mocking, Storage control, New diff editor, Global search, and Auto-pulling of files. A "Continue" button is at the bottom right.
- Screenshot 2:** Promotes "Collaboration with Cloud Sync now available". It compares two options:
  - Store in Local Vault:** Stored locally only, with no cloud. Ideal when collaboration is not needed.
  - Enable Cloud Sync:** Encrypted and synced securely to the cloud, ideal for out-of-the-box collaboration.A "Continue" button is at the bottom right.
- Screenshot 3:** Offers a "Get started for free" call-to-action. It lists sign-in methods:
  - Continue with Google
  - Continue with GitHub
  - Continue with Email
  - Continue with SSOAt the bottom, it includes a note about terms of service and privacy policy, and a "Use the local Scratch Pad" option.

# Insomnia

The screenshot shows the Insomnia application interface. At the top, there is a navigation bar with the title "Insomnia" and a purple circular icon. The menu bar includes "Application", "File", "Edit", "View", "Window", "Tools", and "Help". Below the menu is a search bar with the placeholder "Search..." and a "Ctrl + P" keybinding. On the left side, there is a sidebar titled "Personal Workspace" with a "Home" icon. It lists "PROJECTS (2)" with a "Filter" button and a plus sign. Under "PROJECTS", there are three items: "Personal Workspace", "Insomnia" (selected), and another unnamed project. Below this, there are sections for "All Files (4)", "Documents (1)", "Collections (3)" (which is selected and highlighted in grey), "Mock (0)", and "Environments (0)". In the main workspace, there is a "Filter" input field and a "Create" button. Three collections are listed: "Testes\_Revenda" (4 days ago), "Teste2" (8 days ago), and "Estoque23" (9 days ago). A modal dialog box is open in the center, titled "Create New Request Collection". It contains a text input field with the value "API\_manha\_24". At the bottom right of the dialog is a large purple "Create" button.

# Insomnia

The screenshot shows the Insomnia application interface. At the top, there is a navigation bar with links: Application, File, Edit, View, Window, Tools, and Help. On the right side of the header, there is a purple circular icon with a white circle inside, followed by the word "Insomnia". To the right of that are "Invite" and "edeciofernando@gmail.com".

The main workspace displays a sidebar on the left with a purple circular icon, a search bar labeled "Search.. Ctrl + P", a home icon, and a plus sign icon. Below these are buttons for "Base Environment", "Add Cookies", and "Add Certificates". A "Filter" input field is also present.

A context menu is open in the center-right area, titled "+ CREATE". It contains the following items:

- New Folder Ctrl + Shift + N
- HTTP Request Ctrl + N
- Event Stream Request (SSE)
- GraphQL Request
- gRPC Request
- WebSocket Request

Below this section is a "IMPORT" heading with two options:

- From Curl
- From File

To the right of the menu, there are several buttons with keyboard shortcuts:

- New Request Ctrl + N
- Switch Requests Ctrl + P
- Edit Environments Ctrl + E
- New HTTP Request (highlighted in a red box)

# Insomnia

The screenshot shows the Insomnia API client interface. On the left, a sidebar lists various request types: + CREATE, + HTTP Request (selected), + Event Stream Request (SSE), + GraphQL Request, + gRPC Request, + WebSocket Request, and New Folder. The main area displays a request configuration for a GET endpoint:

- Request Method:** GET
- URL:** https://api.myproduct.com/v1/users
- Buttons:** Send, URL PREVIEW, Import from URL, Bulk Edit
- Params:** A table with columns for name and value, containing one entry: name (name) and value (value).
- Body:** (Empty)
- Auth:** (Empty)
- Headers:** (Empty)
- Scripts:** (Empty)
- Docs:** (Empty)

At the bottom, there are buttons for Filter, Aula1, GET, New Request, and Focus URL.

# Insomnia (GET)

Insomnia

Application File Edit View Window Tools Help

Search.. Ctrl + P

Invite edeciofernando@g

API\_manha\_24 ▾

GET http://localhost:3000/alunos

Send 200 OK 80 ms 149 B

Params Body Auth Headers (3) Scripts Docs Preview Headers (7) Cookies → Mock

No Body

Preview

1 [  
2 {  
3 "id": 1,  
4 "nome": "Alice Santos",  
5 "curso": "ADS"  
6 },  
7 {  
8 "id": 2,  
9 "nome": "Marcelo Pereira",  
10 "curso": "Redes"  
11 },  
12 {  
13 "id": 3,  
14 "nome": "Fabricia Nobre",  
15 "curso": "Marketing"  
16 }  
17 ]

Filter

Aula1

GET Listagem

Enter a URL and send to get a response

Select a body type from above to send data

# Insomnia (POST)

The screenshot shows the Insomnia application interface. In the top navigation bar, there are links for Application, File, Edit, View, Window, Tools, and Help. On the right side of the header, there is an 'Invite' button and a user profile icon for 'edeciofernando@gmail.com'. The main workspace displays a POST request to the endpoint `http://localhost:3000/alunos`. The request details include:

- Method: POST
- URL: `http://localhost:3000/alunos`
- Status: 201 Created
- Time: 95 ms
- Size: 46 B

The request body is set to JSON format, containing the following data:

```
1 {  
2   "nome": "Isabela Santos",  
3   "curso": "Processos Gerenciais"  
4 }
```

The response preview shows:

```
1 {  
2   "id": 4,  
3   "mensagem": "Ok! Inserido com sucesso"  
4 }
```

On the left sidebar, under the environment dropdown 'API\_manca\_24', there are sections for Base Environment, Add Cookies, and Add Certificates. A 'Filter' input field is also present. Below the environment dropdown, there are two items listed: 'POST Inclusão' and 'GET Listagem'. The 'Inclusão' item is highlighted with a green button.

## Route parameters

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys.

```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```

To define routes with route parameters, simply specify the route parameters in the path of the route as shown below.

```
app.get('/users/:userId/books/:bookId', (req, res) => {
  res.send(req.params)
})
```

The name of route parameters must be made up of "word characters" ([A-Za-z0-9\_]).

Since the hyphen (-) and the dot (.) are interpreted literally, they can be used along with route parameters for useful purposes.

```
Route path: /flights/:from-:to
Request URL: http://localhost:3000/flights/LAX-SFO
req.params: { "from": "LAX", "to": "SFO" }
```



CREATE

READ

UPDATE DELETE

# C R U D

```
32
33   router.delete('/:id', (req, res) => {
34     const id = req.params.id
35
36     const index = alunos.findIndex(aluno) => aluno.id === Number(id))
37
38     if (index === -1) {
39       res.status(404).send('Aluno não encontrado')
40     } else {
41       alunos.splice(index, 1)
42       res.status(204).send()
43     }
44   })
45
46   export default router
47
```

# Insomnia (DELETE)

Insomnia

Application File Edit View Window Tools Help

Search.. Ctrl + P

Invite edeciofernando@g

API\_manca\_24

DELETE http://localhost:3000/alunos/2

Send 204 No Content 109 ms 0 B

Base Environment Add Cookies Add Certificates

No Body

Params Body Auth Headers (3) Scripts Docs Preview Headers (4) Cookies → Mock

Preview

No body returned for response

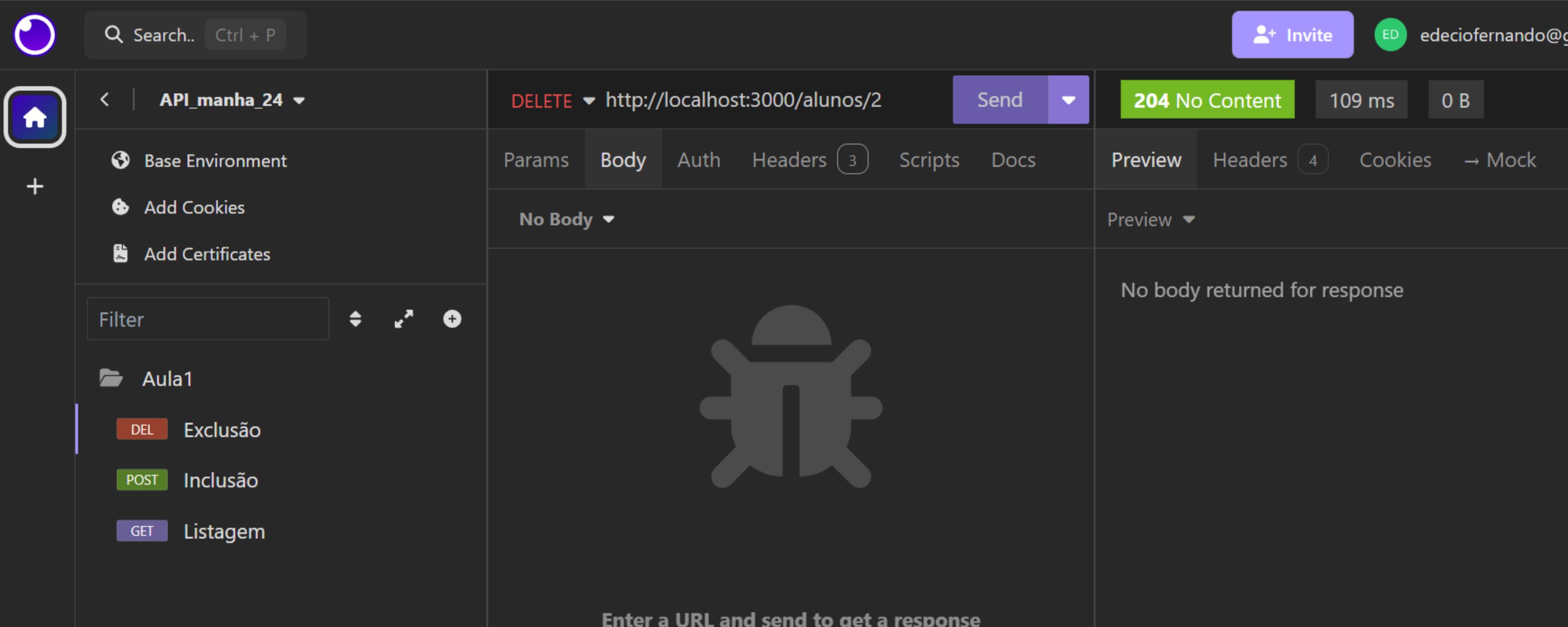
Aula1

DEL Exclusão

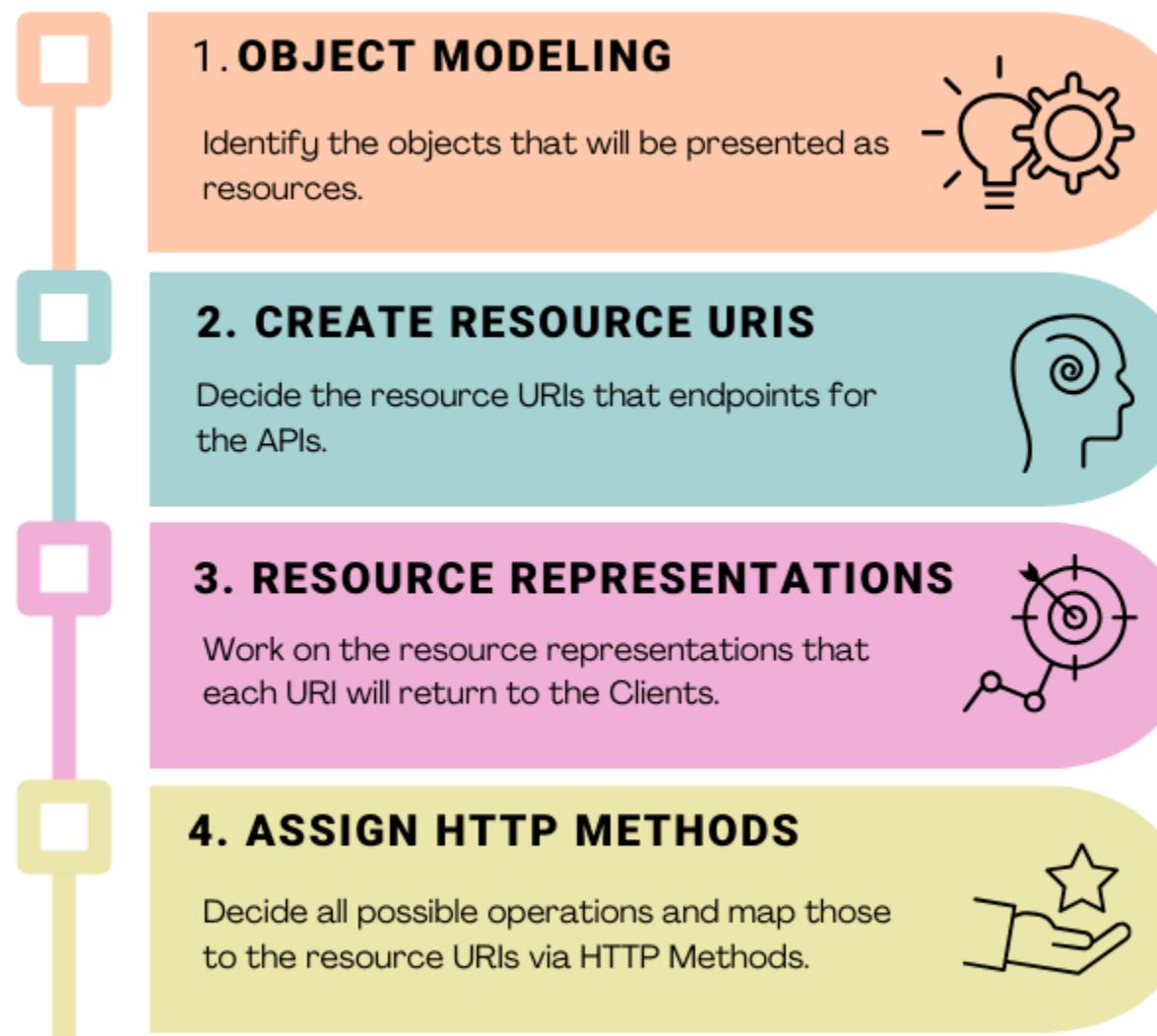
POST Inclusão

GET Listagem

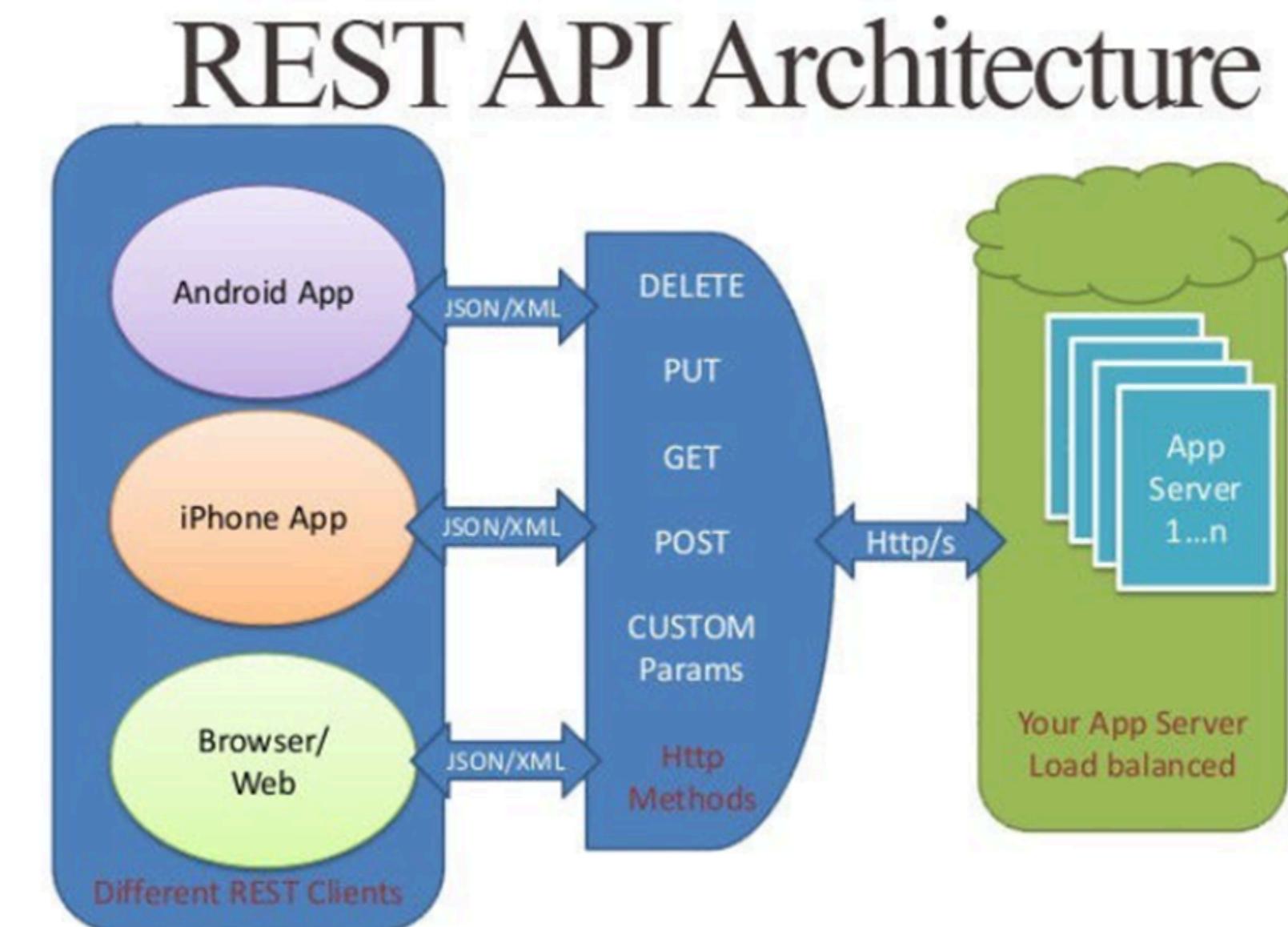
Enter a URL and send to get a response

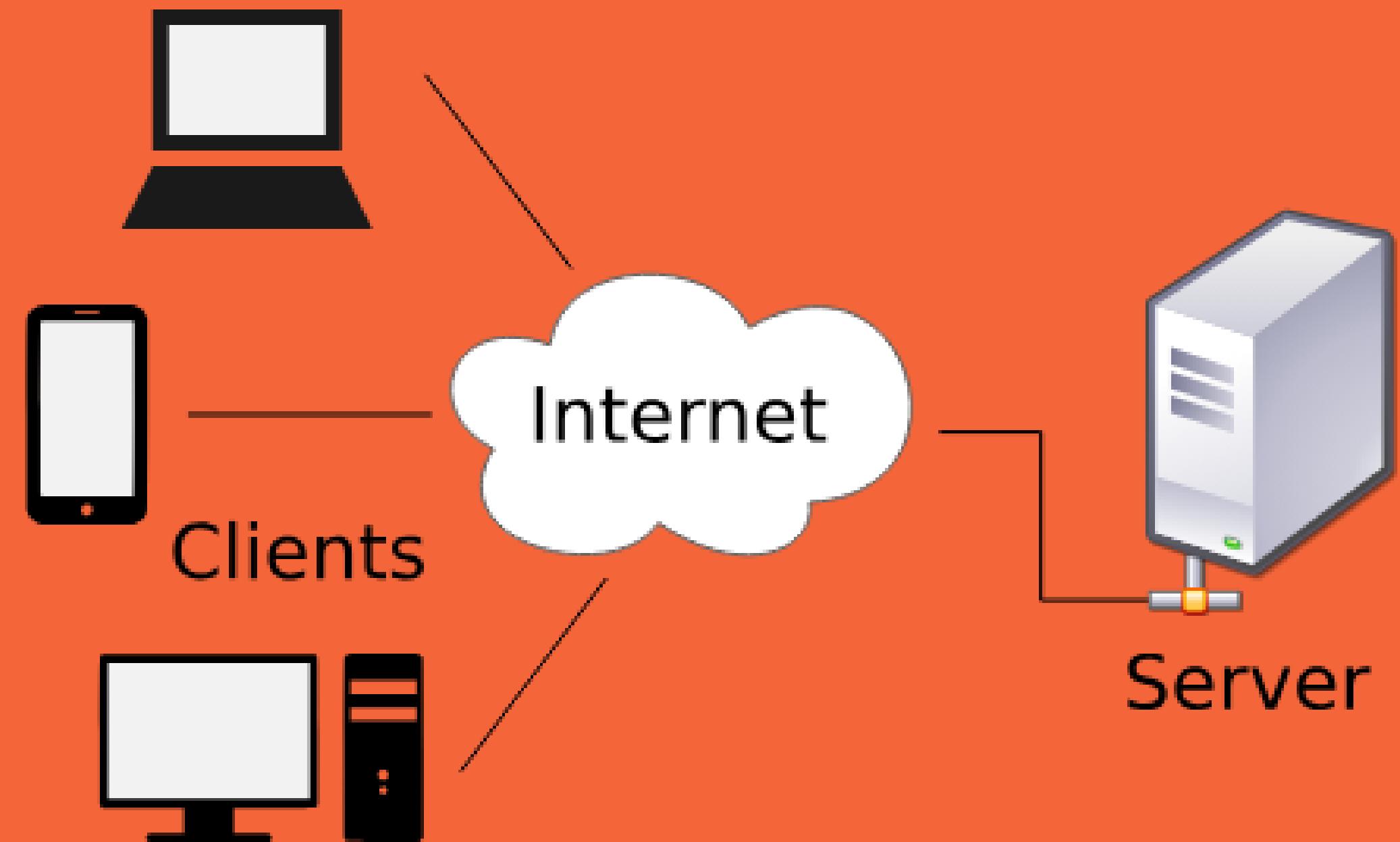


## Steps to Design a REST API



<https://restfulapi.net>





# Modelo Cliente x Servidor

# A seguir...

The screenshot shows the Prisma.io/express website. The URL 'prisma.io/express' is visible in the browser's address bar. The page features a dark blue header with the Prisma logo, navigation links for Products, Pricing, Resources, and Blog, and a search icon. On the right side of the header are 'Login' and 'Get started' buttons. The main content area has a purple background with white wavy patterns. It displays the text 'Easy, type-safe database access in Express servers'. Below this, a subtext explains: 'Query data from MySQL, PostgreSQL & SQL Server databases in Express apps with Prisma – a better ORM for JavaScript and TypeScript.' At the bottom left is a button labeled 'See code examples' with a downward arrow icon. Two white callout boxes are overlaid on the right: one containing the letters 'ex' and another containing a black triangle icon.

prisma.io/express

Prisma Products ▾ Pricing Resources ▾ Blog

>Login Get started

# Easy, type-safe database access in Express servers

Query data from MySQL, PostgreSQL & SQL Server databases in Express apps with Prisma – a better ORM for JavaScript and TypeScript.

See code examples